Fast adaptive real-time classification for data streams with concept drift

Fast Adaptive Real-Time Classification for Data Streams with Concept Drift

Mark Tennant[1], Frederic Stahl[1], João Bártolo Gomes[2]

[1] University of Reading, PO Box 225, Whiteknights, Reading, RG6 6AY, UK,
m.tennant@pgr.reading.ac.uk, F.T.Stahl@reading.ac.uk,
[2] Institute for Infocomm Research (I2R), A*STAR, Singapore, 1 Fusionopolis Way Connexis, Singapore 138632 bartologjp@i2r.a-star.edu.sg

Abstract. An important application of Big Data Analytics is the real-time analysis of streaming data. Streaming data imposes unique challenges to data mining algorithms, such as concept drifts, the need to analyse the data on the fly due to unbounded data streams and scalable algorithms due to potentially high throughput of data. Real-time classification algorithms that are adaptive to concept drifts and fast exist, however, most approaches are not naturally parallel and are thus limited in their scalability. This paper presents work on the Micro-Cluster Nearest Neighbour (MC-NN) classifier. MC-NN is based on an adaptive statistical data summary based on Micro-Clusters. MC-NN is very fast and adaptive to concept drift whilst maintaining the parallel properties of the base KNN classifier. Also MC-NN is competitive compared with existing data stream classifiers in terms of accuracy and speed.

Keywords: Data Stream Classification, Adaptation to Concept Drift, High Velocity Data Streams

1 Introduction

The work presented in this paper focuses on some of the challenges associated with the *velocity* aspect of Big Data [4]. Velocity in Big Data Analytics refers to data instances that arrive at a very high speed and thus challenge our computational capabilities in processing data [6]. Data stream classification trains a classifier in real-time on incoming data instances with a known classification, in order to enable the classification of previously unseen data instances. It is important that the classifier adapts to changes in the pattern encoded in the stream in order to keep the model accurate over time. Such changes in the pattern are also called concept drifts [5]. Some applications of data stream classification include sensor networks; Internet traffic management and web log analysis [8]; intrusion detection [9]. It is not feasible to capture, store and process data streams; as data streams are potentially infinite. Hence, algorithms are needed that can analyse data on the fly as it is being generated. Systems that make use of such algorithms are of great importance to applications such as the ones described above. In the past two decades various data stream classifiers have been

published, such as Hoeffding Trees [3], G-eRules [10], Very Fast Decision Rules (VFDR) [7] etc. These algorithms induce a classifier and adapt to concept drifts with only one pass through the data, making them relatively fast. This research paper proposes a new adaptive computationally efficient data stream classifier. The new classifier proposes a Micro-Cluster based data structure with Variance based splitting. This Micro-Cluster structure is coupled with a K Nearest Neighbour (KNN) classifier approach termed MC-NN. Variance based Micro-Clusters continuously adapt to concept drifts through updating statistical summaries of data instances from the data stream and are robust to noise. KNN has been used as a base classification approach, as KNN is naturally parallel and thus allows for future works to be applied in a parallel framework.

This paper is organised as follows: Section 2 describes the MC-NN algorithm whereas Section 3 provides an empirical evaluation of MC-NN and a comparison against existing data stream classifiers. Conclusions are discussed in Section 4.

2 Adaptive Micro-Cluster Nearest Neighbour Data Stream Classification

2.1 Micro-Cluster based Nearest Neighbour

In the authors' previous feasibility study [12], a parallel real-time classifier was implemented based upon KNN. In KNN a data instance is assigned the class that is most common amongst its K nearest neighbours. The basic approach of the real-time KNN is to keep a sliding fixed size time window of the most recent data instances and execute KNN from the sliding window set. Real-time KNN retrains on recent instances whilst older instances are deleted. However, real-time KNN is computationally slow with faster data streams [12]. To overcome the computational bottleneck of real-time KNN and the problems associated with the sliding window, the here presented classifier adapts Micro-Clusters. Micro-Clusters, originally developed for data stream clustering [1] in order to provide a summary of the locality of the data are of the form:

$$< CF2^x, CF1^x, CF2^t, CF1^t, n >.$$

The sum of the squares of the attributes are maintained the vector $CF2^x$, the sum of the values in vector $CF1^x$; the sum of time stamps in vector $CF1^t$; and the number of data instances is stored in scalar n. $CF2^x$ and $CF1^x$ can be used to calculate the locality and boundary of the Micro-Clusters whereas $CF2^t$ and $CF1^t$ can be used to determine the *recency* of the data summarised in the cluster. MC-NN adapts Micro-Clusters to compute nearest neighbours for classification. The Micro-Cluster structure has been extended by terms CL for the cluster's class label, ϵ as error count, Θ as error threshold for *splitting*, α as initial time stamp and Ω as a threshold for the Micro-Cluster's performance:

$$< CF2^x, CF1^x, CF1^t, n, CL, \epsilon, \Theta, \alpha, \Omega >$$

The centroid of the Micro-Cluster can be calculated by $\frac{CF1^x}{n}$. In order to classify a new data instance from the stream the MC-NN classifier calculates the

Euclidean distances between the data instance and each Micro-Cluster centroid and the class label of the nearest Micro-Cluster is assigned to the data instance. ϵ of a Micro-Cluster is initially 0 and incremented by 1 if the Micro-Cluster is used for classification and missclassifies the data instance. Likewise ϵ is decremented by 1 if the Micro-Cluster is involved in a correct classification. Θ is a user defined upper limit of acceptable ϵ. It is expected that a low Θ will cause the algorithm to adapt to changes faster, but will be more susceptible to noise. A larger Θ value will be more tolerant to noise but may not 'learn' as fast. As more labelled

Algorithm 1: Training the MC-NN classifier

```
Data: Train Instance
Result: Re-Positioned Localised sub-set of Micro-Clusters
Remove Micro-Clusters with poor performance (under Ω value)
foreach Micro-Cluster in LocalSet do
  | Evaluate Micro-Cluster against NewInstance;
end
Sort EvaluationsByDistance();
if Nearest Micro-Cluster is of the Training Items Class Label then
  | CorrectClassification Event
  | NewInstance is Incremented into Nearest Micro-Cluster Nearset Micro-Cluster Error
  | count (ϵ) reduced.
else
  | MisClassification Event
  | 2 Micro-Clusters Identified:
  | 1) Micro-Cluster that should have been identified as the Nearest to the New Instance of
  | the same Classification Label.
  | 2) Micro-Cluster that incorrectly was Nearest the New Instance.
  | Training Item incrementally added to Micro-Cluster of Correct Classification Label.
  | Both Micro-Clusters have internal Error count (ϵ) Incremented
  | foreach Micro-Cluster Identified do
  |   | if Micro-Cluster Error count (ϵ) exceeds Error Threshold (θ) then
  |   |   | Sub-Divide Micro-Cluster upon attribute of largest Variance
  |   | end
  | end
end
```

instances are received for learning they will change the distribution of the Micro-Clusters. According to Algorithm 1 two scenarios are possible after the nearest Micro-Cluster has been identified when a new training instance is presented to the classifier:

Scenario 1: If the nearest Micro-Cluster is of the same label as the training instance, then the instance is incrementally added to the Micro-Cluster and ϵ is decremented by 1.

Scenario 2: If the nearest Micro-Cluster is of a different class label, then the training instance is incrementally added to the nearest Micro-Cluster that matches the training instance's class label. However, the error count ϵ of both involved Micro-Clusters is incremented.

If over time a Micro-Cluster's error count ϵ reaches the error threshold Θ, then the Micro-Cluster is *split*. This is done by evaluating the Micro-Cluster's dimensions for the size of its variance, which can be calculated using Equation (1), where x denotes a particular attribute. The splitting of a Micro-Cluster

generates two new Micro-Clusters, centred about the point of the parent Micro-Cluster's attribute of greatest variance; while the parent Micro-Cluster is removed. The assumption behind this way of splitting attributes is that a larger variance value of one attribute over another indicates that a greater range of values have been seen for this attribute. Therefore the attribute may contribute towards miss-classifications. This splitting of a Micro-Cluster causes the two new Micro-Clusters to separate and better fit the underlying concept encoded in the stream. Once the attribute of largest variance has been identified, the two new Micro-Clusters are initially populated with the parent's internal mean / centre data ($CF1^x$). The split attribute (with the largest variance), is altered by the variance value identified in the positive direction in one of the new Micro-Clusters and negatively in the other. This ensures that future training will further re-position the two new Micro-Clusters better than the parent could alone.

$$Variance[x] = \sqrt{\left(\frac{CF2^x}{n}\right) - \left(\frac{CF1^x}{n}\right)^2} \quad (1)$$

When a Micro-Cluster has a new instance added to it, it's internal instance count n is incremented by 1 and the sum of time stamps($CF1^t$) is incremented by the new time stamp value(T). The *Triangle Number* $\Delta(T) = ((T^2 + T)/2)$ of this time stamp will give an upper bound to the maximum possible value of $CF1^t$. Therefore, if all instances were entered into this Micro-Cluster $CF1^t$ would be equal to the triangular number of T. The lower the value of $CF1^t$ is from the *Triangular Number* the poorer the Micro-Cluster has been participating in the stream classification. The use of Triangular Numbers give more importance to recent instances over earlier ones added to the Micro-Cluster, as the time stamp value (T) is always increasing and MC-NN uses the sum of these incremental values. Triangular numbers assume that all Micro-Clusters were created at time stamp 1. To counter this each Micro-Cluster keeps track of the time stamp when it was initialised (α). The Micro-Cluster's real $\Delta(T)$ can be calculated by $\Delta(T) - \Delta(\alpha)$. Any Micro-Clusters that fall under a pre-set threshold value of (Ω) are deleted as they are considered old. For the rest of this paper a value of 50% was given to all Micro-Cluster Ω values as it seemed to work best for most classification problems.

3 Evaluation

This Section evaluates MC-NN in terms of accuracy, adaptivity to concept drifts and computational efficiency on a quad core 'Intel core' I5 processor with 8Gb RAM. All classifiers and data stream generators are implemented in the Massive Online Analysis (MOA) framework. Three data streams have been utilised: The **SEA data stream** [11] contains three continuous attributes and two class labels. A class label of *True* is given only if the Threshold level of a preset value is surpassed by summing two of the attributes, otherwise class label *False* is given. Arbitrarily function 1 (value 8) was chosen for the initial concept and function

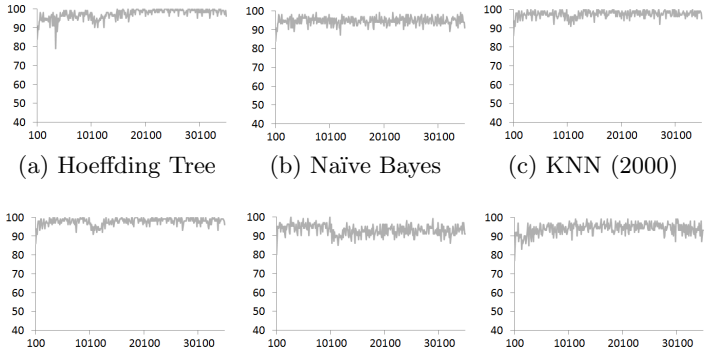
3 (value 7) for the concept change. The **Random Tree Generator** [2] creates a random tree with each leaf node randomly assigned a class label. In our experiments the random tree(s) comprise ten continuous attributes and three class labels. A drift is achieved by simply generating a different random tree. Both, the Random Tree and the SEA datastreams generated 35,000 instances. The concept drift begins at instance 10,000 with a gradual change over 1,000 instances to the second stream. The **Hyperplane generator** creates a linearly separable model. A Hyperplane in 'D' dimensions slowly rotates continuously changing the linear decision boundary of the stream. The experiments using the Hyperplane generator created 10 million data instances, with five numerical attributes and two classes. In order to add an additional challenge 10% noise was generated as well with probability P(0.75) chance of reversing the direction of the rotation causing an 'Oscillation' effect. A version of the stream with probability P(0) chance of reversing the direction of the concept drift was also created. MC-NN was compared against Hoeffding Trees [3], incremental Naïve Bayes and real-time KNN classifier [12]. Each instance was tested upon the classifier to log the classifier's performance before being used for training: this is also know as prequential testing.

Adaptation to new concepts: Two MC-NN classifiers were created, one with $\Theta = 2$ (error threshold) and the other with $\Theta = 10$. Table 1 compares MC-NN against other stream classification algorithms on the SEA and Random Tree data streams. Please note that for real-time KNN several experiments have been carried out and only the experiments with the best setting for K are included in the table. The results show that real-time KNN's results are competitive to the Hoeffding Tree and Naïve Bayes classifiers. MC-NN achieves accuracies close to all competitors, while clearly outperforming real-time KNN in terms of runtime. Regarding accuracy MC-NN is similar to Hoeffding Trees and Naïve Bayes. It is also noticeable that a larger Θ results in a shorter runtime of MC-NN. This can be explained by the fact that when Θ is larger it will take more time for a Micro-Cluster to reach Θ and thus it will perform splits less frequently.

Table 1: Accuracies and runtime of MC-NN compared with other data stream classifiers. Accuracies are listed in percent and runtime is listed in seconds. Θ denotes the error threshold used in MC-NN

Algorithm	SEA accuracy(runtime)	Random Tree accuracy(runtime)
Naïve Bayes	94.40(0.11)	64.17(0.10)
Hoeffding Tree	95.96(0.19)	69.88(0.28)
real-time KNN	97.17(24.73) K=5000	71.34(9.04) K=2000
MC($\Theta = 2$)	94.03(0.28)	70.30(2.02)
MC($\Theta = 10$)	92.99(0.03)	60.99(1.49)

Figures 1 and 2 illustrate the same experiments as listed in Table 1, the accuracy is displayed over time. For SEA it can be seen that all classifiers achieve a relatively high accuracy at any time and only show a slight deterioration in



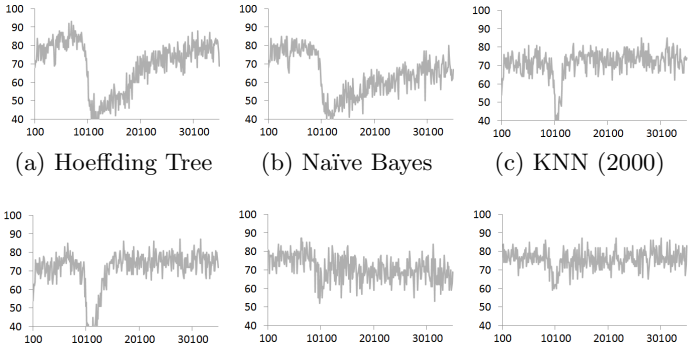
(a) Hoeffding Tree (b) Naïve Bayes (c) KNN (2000)

(d) KNN (5000) (e) Micro-Cluster(2) (f) Micro-Cluster(10)

Fig. 1: Concept drift adaptation on the SEA data stream. Accuracy is plotted along the vertical axis, instance stream is plotted along the horizontal axis.

accuracy during the concept drift (instances 10,000 - 11,000). For the Random Tree it can be seen that Hoeffding Tree and Naïve Bayes classifiers are clearly challenged with adapting to the concept drift as they need a long time to fully regain their previous classification accuracy level. The real-time KNN classifer also have a noticeable deterioration of their classification accuracy during the concept drift but recover much faster compared with Hoeffding Tree and Naïve Bayes. However, they do not reach the same level of classification accuracy as Hoeffding trees and Naïve Bayes. The results of MC-NN clearly show the lowest classification accuracy deterioration and almost recover instantly. MC-NN is able to reach the same classification accuracy levels as Hoeffding tree and Naïve Bayes, whereas real-time KNN performs poorly.

The Results in Figures 3 and 4 show the total accuracy of the different classifiers evaluated on the Hyperplane data streams with their runtime in brackets. In terms of classification accuracy it can be seen that MC-NN(10) achieves second highest accuracy, but only 0.04% behind Naïve Bayes on the stream with no oscillation. On the the stream with oscillation effect MC-NN(10) clearly outperforms all its competitors. Please note that the Figures display only the runtime for the best configurations with real-time KNN. In terms of runtime, MC-NN is faster than Hoeffding Trees and achieves a similar speed to that of Naïve Bayes. However, MC-NN is approximately 30 times faster than real-time KNN. Please note that for the larger Θ MC-NN performs slightly faster, which can be explained by MC-NN being less likely to perform Micro-Cluster splits which consume some of the runtime. Figure 3 shows the experiments for the Rotating Hyperplane data stream over time for all 10 million data instances. All classifiers need some initialisation phase before producing a stable classification accuracy. Overall MC-NN(10) achieves a similar performance to Naïve Bayes and outperforms is predecessor real-time KNN clearly. Figure 4 shows the experiments for the Oscillating Hyperplane data stream over time for all 10 million data in-



(a) Hoeffding Tree (b) Naïve Bayes (c) KNN (2000)

(d) KNN (5000) (e) Micro-Cluster(2) (f) Micro-Cluster(10)

Fig. 2: Concept drift adaptation on the Random Tree data stream. Accuracy is plotted along the vertical axis, instance stream is plotted along the horizontal axis.

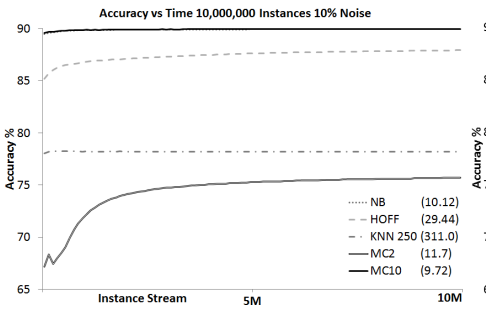


Fig. 3: Concept Drift adaptation on the Hyperplane with Rotating Boundary.

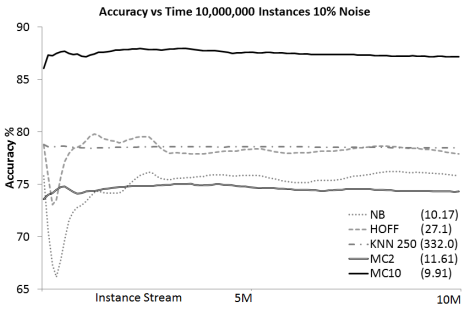


Fig. 4: Concept Drift adaptation on the Hyperplane with Oscillating Boundary.

stances. MC-NN(10) remains stable and clearly outperforms all its competitors. Both Naïve Bayes and the Hoffeding Tree classifiers suffer at the beginning of the data stream with a negative accuracy trend. This is due to the overlapping data values that are contradicting each other due to oscillation. Overall MC-NN achieves a similar performance compared with well established data stream classifiers in terms of accuracy and runtime and clearly outperforms its predecessor. MC-MM is more robust in terms of adaptation to concept drifts, especially complex continuous concept drifts. Moreover MC-NN is naturally parallel and thus has the advantage to be scaled up to high speed data streams.

4 Conclusions

This paper presents the development of the novel MC-NN data stream classifier that is competitive with popular existing data stream classifiers in terms of ac-

curacy and adaptability to concept drifts, but is also computationally efficient and potentially scalable to parallel computer architectures. The developed classifier is based on a nearest neighbour approach for classification and on a novel kind of Micro-Cluster for classification purposes to maintain a recent summary of the data observed and its performance. MC-NN has been compared empirically with Hoeffding tree, Naïve Bayes for streaming data and its predecessor real-time KNN. Empirical results show that MC-NN achieves similar or better accuracy, adaptability to concept drifts and shorter runtime compared with its competitors. Notably MC-NN is very robust when confronted with continuously changing concepts and noise. The paper also points out that MC-NN is naturally parallel as Micro-Clusters can be distributed over multiple computational nodes in a computer cluster. Therefore ongoing work comprises the implementation and empirical evaluation of a new parallel MC-NN classifier.

References

1. C. Aggarwal, J. Han, J. Wang, and P.Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th VLDB Conference*, Berlin Germany, 2003.
2. P Domingos and G Hulten. Mining high-speed data streams. *KDD*, pages 71–80, 2000.
3. Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pages 71–80, New York, NY, USA, 2000. ACM.
4. M Ebbers, A Abdel-Gayed, V Budhi, and F Dolot. *Addressing Data Volume, Velocity, and Variety with IBM InfoSphere Streams V3.0*. 2013.
5. M. Gaber, A. Zaslavsky, and S. Krishnaswamy. A survey of classification methods in data streams. *Data Streams*, pages 39–59, 2007.
6. Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. Mining data streams: a review. *ACM SIGMOD Record*, 34:18–26, June 2005.
7. João Gama and Petr Kosina. Learning decision rules from data streams. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, pages 1255–1260. AAAI Press, 2011.
8. João Gama. *Knowledge Discovery from Data Streams*. Chapman and Hall / CRC, 2010.
9. A. Jadhav, A. Jadhav, P. Jadhav, and P. Kulkarni. A novel approach for the design of network intrusion detection system(NIDS). In *Sensor Network Security Technology and Privacy Communication System (SNS PCS), 2013 International Conference on*, pages 22–27, May 2013.
10. Thien Le, Frederic Stahl, João Bártolo Gomes, Mohamed Medhat Gaber, and Giuseppe Di Fatta. Computationally efficient rule-based classification for continuous streaming data. In *Research and Development in Intelligent Systems XXXI*, pages 21–34. Springer International Publishing, 2014.
11. W.N. Street and Y.S. Kim. A streaming ensemble algorithm (SEA) for large-scale classification. *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 377–382, 2001.
12. Mark Tennant, Frederic Stahl, Giuseppe Di Fatta, and JooBrtolo Gomes. Towards a parallel computationally efficient approach to scaling up data stream classification. In Max Bramer and Miltos Petridis, editors, *Research and Development in Intelligent Systems XXXI*, pages 51–65. Springer International Publishing, 2014.